

CPE

Comandos de Entrada e Saída

Departamento de Engenharia Elétrica – UnB

- 1 Fluxos (Streams)
- 2 Saída de dados
- 3 Entrada de dados

Um parêntese: comentários

O código fonte pode conter comentários direcionados unicamente ao programador. Estes comentários devem estar delimitados pelos símbolos `/*` e `*/` ou `//`, e são ignorados pelo compilador.

- Comentários são úteis para descrever o algoritmo usado e para explicitar suposições não óbvias sobre a implementação.

Exemplo

```
#include <iostream>

/* Este é o meu primeiro programa. */
int main() {
    //'\\n' também representa quebra de linha
    std::cout << "Hello, world!\\n";
    return 0;
}
```

Um parêntese: indentação

Além de colocar comentários, é importante que todo programador faça a indentação do código para que ele fique mais fácil de ler.

- Sem indentação

```
#include <iostream>

int main(){
int i,n;
float nota,media=0.0;
std::cin >> n;
for(i=0;i<n;i++){
std::cout << "Digite a " << i+1 << "a nota: ";
std::cin >> nota;
media = media + nota;
}
media = media/n;
std::cout << "Media: " << media;
return 0;
}
```

- Com indentação

```
#include <iostream>

int main(){
    int i,n;
    float nota,media=0.0;

    std::cin >> n; //leitura do numero de notas
    for(i=0;i<n;i++){
        std::cout << "Digite a " << i+1 << "a nota: ";
        std::cin >> nota; //leitura dos valores
        media = media + nota;
    }
    media = media/n; //media calculada
    std::cout << "Media: " << media;
    return 0;
}
```

Um parêntese: comando using

A linguagem C++ permite definir *namespaces*

- Por exemplo, as variáveis, tipos e objetos disponíveis pela biblioteca padrão tem o nome sempre precedido por `std::`
- Para abreviar as referências a elementos de um determinado namespace, podemos usar o comando `using`

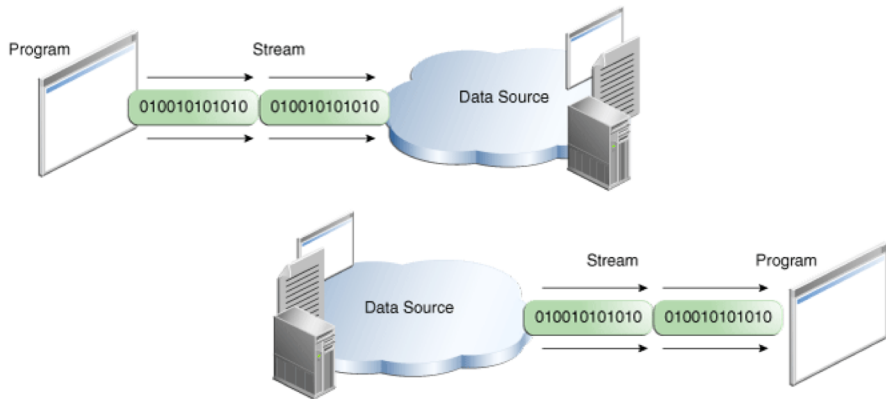
Exemplo

```
using namespace std;
```

Permite por exemplo chamarmos no código só de `cout`, ao invés de `std::cout`

Stream em C++

- Streams ou Fluxos são uma **abstração** que a linguagem C++ oferece para realizarmos operações de entrada (leitura de dados) e saída (escrita de dados)



Streams

- Um stream é uma entidade onde o programa pode mandar escrever/ler caracteres de forma **sequencial**.
- Não é necessário saber detalhes do meio em que isso está acontecendo, isto é, se é o teclado, o terminal, um arquivo, etc.
- Só é necessário saber se o stream é de entrada (leitura) ou de saída (escrita).

Streams da biblioteca padrão do C++

- A biblioteca padrão define alguns objetos stream que são considerados as fontes e destinos “padrão” de caracteres, em um programa:

<code>cin</code>		stream de entrada padrão
<code>cout</code>		stream de saída padrão
<code>cerr</code>		stream de erro padrão (saída)
<code>clog</code>		stream de logging padrão (saída)

- Dedicaremos atenção, por enquanto, somente aos streams `cout` e `cin`, respectivamente a saída padrão (terminal ou prompt) e a entrada padrão (teclado)

Escrevendo o conteúdo de uma variável na tela

- Primeiro, o programa deve começar com a diretiva de inclusão da biblioteca padrão de Input/Output do C++:

```
#include <iostream>
```

- Além de constantes e texto puro, podemos imprimir na **saída padrão** o conteúdo de uma variável utilizando a operação de inserção (<<) seguida do que se deseja imprimir.
- À esquerda do operador <<, indica-se o stream de destino.

```
stream_saída << Expressão;
```

Escrevendo o conteúdo de uma variável na tela

- No caso da saída padrão, o objeto de destino é `std::cout`, que significa **s**tandard **c**haracter **o**utput device
 - Ou somente `cout`, caso tenha colocado antes o comando `using namespace std;`

Exemplo

```
cout << "Frase Saida"; // imprime Frase Saida na tela
cout << 120; // imprime numero 120 na tela
cout << x; // imprime o valor de x na tela
```

Escrevendo o conteúdo de uma variável na tela

- Se queremos imprimir constantes, como uma string, deve-se tomar cuidado para colocar sempre entre aspas duplas

Exemplo

```
cout << "Hello"; // imprime Hello  
cout << Hello; // imprime conteúdo da variável Hello
```

Encadeamento de inserções

Podemos encadear operações de inserção, a ordem de avaliação das operações é sempre da esquerda para a direita.

- Lembre também que todo final de comando deve se encerrar com ;

Exemplo

```
cout << "Isto" << " e um " << "unico comando C++";
```

Encadeamento de inserções

- O encadeamento é muito útil principalmente quando queremos combinar a impressão de constantes e variáveis

Exemplo

```
cout << "Eu tenho " << age << " anos e meu CEP e " <<  
zipcode;
```

Se age contém o valor 24 e zipcode contém o valor 90064, o que será impresso?

Quebra de linha

A escrita em `cout` não leva a quebra-de-linhas, a menos que façamos de forma explícita, usando o caractere especial `'\n'`

Exemplo

```
cout << "First sentence.\n";  
cout << "Second sentence.\n Third sentence.";
```

Em quantas linhas serão impressos os comandos acima?

Quebra de linha

Alternativamente, podemos usar o manipulador `std::endl`, que produz uma quebra de linha mas também **esvazia o buffer** do stream, isto é, obriga a escrever os dados de saída imediatamente, se não tiver ocorrido.

Exemplo

```
cout << "First sentence." << endl;  
cout << "Second sentence." << endl;
```

Ponto flutuante

A impressão de valores de ponto flutuante pode ter o número de casas decimais definido manualmente, através dos manipuladores

`std::setprecision` e `std::fixed`

- Deve-se incluir a diretiva `#include <iomanip>` no começo do programa.

Exemplo

```
cout << std::setprecision(3) << std::fixed;  
cout << "A media final e: " << media << endl;
```

Na saída o conteúdo de `media` será impresso com 3 casas decimais

Ponto flutuante

Se desejamos definir somente o número de dígitos significativos, só usamos o manipulador `std::setprecision`

Exemplo

```
cout << setprecision(3);  
cout << "A media final e:  " << media << endl;
```

Define que a variável `media` será impressa, em geral, com 2 casas à esquerda e 1 à direita da vírgula

Outros caracteres especiais

Exemplo

```
cout << "Caracteres \\ \' \" ";
```

imprime na saída

```
Caracteres \ ' "
```

Entrada de dados do teclado

A entrada padrão (standard input) em geral é associada ao **teclado**.

- O objeto stream que a representa é `std::cin`, que significa **standard character input device**
- A operação de leitura do stream então é feita usando o operador de **extração** `>>` seguido da variável que armazenará o conteúdo lido

Exemplo

```
int age;  
cin >> age;
```

Entrada de dados do teclado

- A operação `cin >> age;` faz o programa esperar por alguma entrada vinda de `cin`; isso significa na prática que o programa irá esperar o usuário digitar alguma sequência até ocorrer uma quebra de linha (ENTER), um espaço ou fim de arquivo (End-Of-File).
- O **tipo da variável** após o operador `>>` é que determina como interpretar os símbolos lidos da entrada
 - Se for inteiro, espera-se uma sequência de algarismos
 - Se for ponto flutuante, espera-se sequência de algarismos, possivelmente com valores após o `'.'`

Entrada de dados - exemplo

```
1 //CPE – Exemplo de Entrada / Saida
2 #include <iostream>
3 using namespace std;
4
5 int main (){
6     int i;
7     cout << "Entre com um valor inteiro: ";
8     cin >> i;
9     cout << "O valor fornecido e " << i;
10    cout << " e o seu dobro e " << i*2 << ".\n";
11    return 0;
12 }
```

E se não digitarmos um inteiro? O comando >> não extrai um valor para i e o programa continua com um comportamento imprevisível

- Mais adiante veremos formas de contornar isso...

Encadeamento de extrações

Podemos ler múltiplos valores em um só comando.

- Por exemplo, ler do teclado 2 valores para armazenar nas variáveis a e b:

```
cin >> a >> b;
```

o que equivale fazer

```
cin >> a;
```

```
cin >> b;
```

- No caso, espera-se que o usuário digite dois valores para o programa, separados por ENTER ou espaço

Dilbert

